



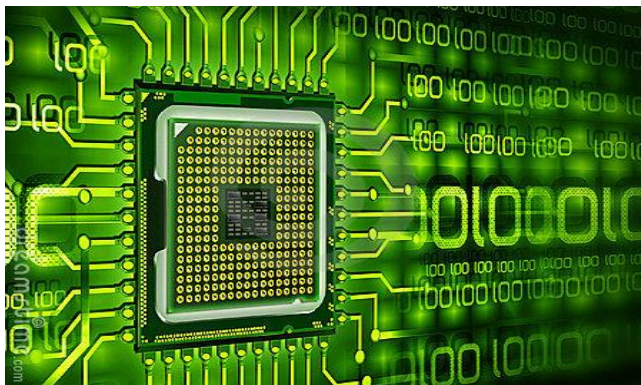
R 1.03

2023 - 2024

Introduction à l'architecture des

Corrigé du TD n° 3

**Codage de l'information
Codes détecteurs d'erreurs**



ANNE Jean-François
D'après le cours de M. JEANPIERRE

Le but de ce TD est de se familiariser avec les codes de détection, voire de correction, d'erreurs

A. Distance de Hamming :

1°) Quelle est la distance de Hamming de ce code binaire (4 mots de 5 bits) ?

00000 ; 01010 ; 10101 ; 11111

Entre le mot 1 et le mot 2, il y a 2 bits de différence => la distance sera au maximum de 2.

On montre par énumération qu'il n'y a pas de distance plus faible.

La distance de Hamming de ce code est donc de 2, car la distance minimum est de 2 bits.

2°) Proposez un code binaire de 10 mots ayant une distance de Hamming de 3 bits.

Il faut donner des mots qui ont 3 bits de différences entre tous :

000000000000 ; 000000000111 ; 000000111000 ; 000111000000 ;
 000111111000 ; 000111111111 ; 111000000000 ; 111000000111 ;
 111000111000 ; 111111000000 ;

B. Code à parité

1°) Quelle est la parité de ce code (8 mots de 4 bits) ?

0 : 0001 ; 1 : 0010 ; 2 : 0100 ; 3 : 0111 ; 4 : 1000 ; 5 : 1011 ; 6 : 1101 ; 7 : 1110

Chaque mot contient un nombre impair de "1". Donc c'est une parité impaire.

2°) Quelle est la distance de Hamming du code précédent ?

Entre le code de 0 et le code de 1, il y a 2 bits de différence. On ne peut pas trouver moins... donc la distance de Hamming est de 2.

3°) Codez les nombres suivants avec le code précédent :

123₈ => 0010 0100 0111 ;

1^{ère} solution :

001010011₂ => 0001 0001 0010 0001 0010 0001 0001 0010 0010₂ ;

123₁₆ => 0010 0100 0111₁₆ ;

12₁₆ => 0010 0100₁₆ ;

2^{ème} solution :

001010011₂ => 123₈ => 0010 0100 0111 ;

123₁₆ => 000100100011₂ => 0443₈ => 0001 1000 1000 0111 ;

$12_{16} \Rightarrow 00010010_2 \Rightarrow 022_8 \Rightarrow 0001\ 0100\ 0100 ;$

4°) Codez le message (1234₈) avec le code précédent

$1234_8 \Rightarrow 0010\ 0100\ 0111\ 1000$

5°) Quelle est la distance de Hamming du message obtenu ?

$D(1,2) = 2 ; D(2,3) = 2 ; D(3,4) = 4 ; \dots$

La distance de Hamming est de 2.

6°) Le mettre sous la forme d'un code de bloc, en ajoutant un contrôle de parités horizontale et verticale impaires et un contrôle de parité croisée paire.

	VRC					Nb.
	0	0	0	1	0	1
	0	0	1	0	0	2
	0	0	1	1	1	3
	0	1	0	0	0	4
LRC	0	0	1	1	0	

1°) Quelle est la distance de Hamming du message obtenu ?

$D(1,2) = 2 ; D(2,3) = 2 ; D(3,4) = 4 ; D(4,LRC) = 3 ; D(1,LRC) = 1 \dots$

La distance de Hamming est de 1.

C. Contrôle de Redondance Cyclique (CRC)

1°) Soit le polynôme générateur $G(X) = X^5 + X + 1$

a. Quelle sera la longueur du message correspondant à une donnée de :

9 bits $\Rightarrow G(X)$ est de degré 5, il faut d'ajouter 5 bits soit : 14 bits ;

10 bits $\Rightarrow G(X)$ est de degré 5, il faut d'ajouter 5 bits soit : 15 bits ;

1024 octets $\Rightarrow G(X)$ est de degré 5, il faut d'ajouter 5 bits par octet, soit : 1024 octets $\Rightarrow 8 * 1024 + 5$ bits, soit 8197 bits de message ; soit 1025 octets.

b. Déterminer le message binaire à transmettre correspondant à la donnée octale 456₈.

$456_8 \Rightarrow 100101110_2$

Pour le calcul du CRC, on ajoute n bits à 0 à la donnée si le polynôme est de degré n. On effectue les XOR, on obtient un reste sur n bits, que l'on rajoute comme CRC à la fin des données.

10010111000000

100011

000110110

100011

0101010

100011

00100100

100011

00011100

456₈ => 10 010 111 011 100_{CRC} => 22734₈

- c. Vous recevez le message « 76543₈ », sur 15 bits. Pouvez-vous calculer la donnée initiale ?

76543₈ => 111 110 101 100 011_{CRC}

111110101100011

100011

0111011

100011

0110000

100011

0100111

100011

000100100

100011

000111011

100011

011000

Le reste est différent de 0, donc il y a eu erreur de transmission. => On ne peut rien dire sur le message original !

2°) Soit le polynôme générateur $G(X) = X^2 + X + 1$

- a. Vous recevez le message $T(X) = X^{10} + X^7 + X^6 + X^4 + X^3 + X$. Est-il correct ?

$T(X) = X^{10} + X^7 + X^6 + X^4 + X^3 + X$.

T=10011011010

$$G(X) = X^2 + X + 1$$

$$G=111$$

On calcule le reste :

1001101101000

111

0111

111

000101

111

0101

111

0100

111

0111

111

000000

Le message est correct !

- b. Calculez la donnée initiale binaire de ce message T(X)

On supprime les 2 bits du CRC : T=100110110|10

=> M=100110110

- c. Pouvez-vous trouver une altération discrète de ce message (message faux avec CRC juste) ?

Pour altérer discrètement le message il suffit de modifier 3 bits consécutifs (multiple du nombre de bits de G)

T =10011011010

T'=01111011010

01111011010

- 111

1011010

- 111

0101010

- 111

010010

```
- 111
  01110
- 111
  0000
```

Le CRC est juste bien que le message soit faux...

D. Code correcteur d'erreurs

Le protocole Wifi 802.11b définit un codage du bit 1 par 10110111000_2 , et le codage du bit 0 par 01001000111_2 . Ce code, dit DSSS offre une très bonne résistance aux erreurs.

- a. Quelle est sa distance de Hamming ?

```
10110111000
⊕ 01001000111
11111111111
```

DSSS utilise une séquence de Barker de 11 bits. Donc si deux messages diffèrent de 1 bit, leurs codages diffèrent de 11 bits donc la distance de Hamming de ce code est de 11.

- b. Quelle est la donnée initiale de ce message reçu ?

```
11000111000010010001111011011111100110111000010010001111001011100001
00100011110110111001
11000111000|01001000111|10110111111|00110111000|01001000111|10010111000|
01001000111|10110111001
```

Il faut découper le message en blocs de 11 bits, et rechercher le maximum de correspondances avec les mots du code :

```
11000111000 = 1 (7/11)
01001000111 = 0 (11/11)
10110111111 = 1 (8/11)
00110111000 = 1 (10/11)
01001000111 = 0 (11/11)
10010111000 = 1 (10/11)
01001000111 = 0 (11/11)
10110111001 = 1 (10/11)
==> M = 101101012
```

Quelle serait la représentation binaire de la donnée B5₁₆ ?

```
T=B516 = 101101012
1 => 10110111000
```

0 => 01001000111

1 => 10110111000

1 => 10110111000

0 => 01001000111

1 => 10110111000

0 => 01001000111

1 => 10110111000

=>T=101101110000100100011110110111000101101110000100100011110
1101110000100100011110110111000

❖ Opérations simples

○ Non

Effectuez l'opération logique NON (NO) (!) avec les exemples suivants :

00001111₂ => 11110000₂ ;

123₁₀ => 01111011₂ => 10000100₂ => 132₁₀ ;

FA₁₆ => 11111010₂ => 00000101₂ => 05₁₆ ;

255₁₀ => 11111111₂ => 00000000₂ => 0₁₀ ;

○ ET

Effectuez l'opération logique ET (AND) (&) avec les exemples suivants :

$$\begin{array}{r} 1011 \ 0111 \\ \& \ 1101 \ 1111 \\ \hline 1001 \ 0111 \end{array}$$

$$\begin{array}{r} 1100 \ 1011 \\ \& \ 1001 \ 0110 \\ \hline 1000 \ 0010 \end{array}$$

$$\begin{array}{r} 0110 \ 1111 \\ \& \ 1110 \ 1110 \\ \hline 0110 \ 1110 \end{array}$$

100₂ ET 11111101₂ => 100₂

11101010₂ ET 11011100₂ => 11001000₂

5₁₆ ET FB₁₆ => 1₁₆

8₁₆ AND D₁₆ => 8₁₆

7D₁₆ ET (! 73₁₆) => C₁₆

○ OU

Effectuez l'opération logique OU (OR) (|) avec les exemples suivants :

$$\begin{array}{r} 1010 \ 1101 \\ | \ 1001 \ 1111 \\ \hline 1011 \ 1111 \end{array}$$

$$\begin{array}{r} 1010 \ 0100 \\ | \ 0100 \ 1001 \\ \hline 1110 \ 1101 \end{array}$$

$$\begin{array}{r} 1101 \ 1011 \\ | \ 0111 \ 1111 \\ \hline 1111 \ 1111 \end{array}$$

$$101101_2 \text{ OU } 11001100_2 \Rightarrow 11101101_2$$

$$11100110_2 \text{ OU } 10010100_2 \Rightarrow 11110110_2$$

$$9_{16} \text{ OR } 0_{16} \Rightarrow 9_{16}$$

$$A8_{16} \text{ OU } 7E_{16} \Rightarrow FE_{16}$$

$$7D_{16} \text{ OU } (! 73_{16}) \Rightarrow FD_{16}$$

○ **OU Exclusif**

Effectuez l'opération logique OU Exclusif (XOR) (^) avec les exemples suivants :

$$\begin{array}{r} 1010 \ 1101 \\ \wedge \ 1001 \ 1111 \\ \hline 0011 \ 0010 \end{array}$$

$$\begin{array}{r} 1010 \ 0100 \\ \wedge \ 0100 \ 1001 \\ \hline 1110 \ 1101 \end{array}$$

$$\begin{array}{r} 1101 \ 1011 \\ \wedge \ 0111 \ 1111 \\ \hline 1010 \ 0100 \end{array}$$

$$1001010_2 \text{ XOR } 11011011_2 \Rightarrow 10010001_2$$

$$D_{16} \text{ XOR } 3_{16} \Rightarrow E_{16}$$

$$5_{16} \text{ XOR } E_{16} \Rightarrow B_{16}$$

$$4B_{16} \text{ XOR } 12_{16} \Rightarrow 59_{16}$$

$$7D_{16} \text{ XOR } (! 73_{16}) \Rightarrow F1_{16}$$

○ **NON ET**

$$100_2 \text{ NON ET } 11111101_2 \Rightarrow 11111011_2$$

$$11101010_2 \text{ !ET } 11011100_2 \Rightarrow 00110111_2$$

$$5_{16} \text{ !ET } FB_{16} \Rightarrow FE_{16}$$

$$8_{16} \text{ !AND } D_{16} \Rightarrow 7_{16}$$

$$7D_{16} \text{ NAND } (! 73_{16}) \Rightarrow F3_{16}$$

○ **NON OU**

$$101101_2 \text{ NON OU } 11001100_2 \Rightarrow 00010010_2$$

$$11100110_2 \text{ !OU } 10010100_2 \Rightarrow 00001001_2$$

$$9_{16} \text{ !OR } 0_{16} \Rightarrow 6_{16}$$

$$A8_{16} \text{ !OU } 7E_{16} \Rightarrow 01_{16}$$

$$7D_{16} \text{ NOR } (! 73_{16}) \Rightarrow 02_{16}$$

○ **NON OU Exclusif**

$1001010_2 \text{ XOR } 11011011_2 \Rightarrow 01101110_2$

$D_{16} \text{ XOR } 3_{16} \Rightarrow 1_{16}$

$5_{16} \text{ XOR } E_{16} \Rightarrow 4_{16}$

$4B_{16} \text{ XOR } 12_{16} \Rightarrow A6_{16}$

$7D_{16} \text{ XOR } (! 73_{16}) \Rightarrow 0E_{16}$

❖ Décalages

○ A gauche

❖ Que devient le nombre hexadécimal D après un décalage à gauche de 2 bits ?

$D \ll 2 = 34$

❖ Que devient le nombre hexadécimal 5 après un décalage à gauche de 2 bits ? :

$5 \ll 2 = 14$

❖ Que devient le nombre hexadécimal D5 après un décalage à gauche de 1 bit ?

$D5 \ll 1 = 1AA$

❖ Que devient le nombre hexadécimal AAAA après un décalage à gauche de 4 bits ?

$AAAA \ll 4 = AAAAA0$

○ A droite

❖ Que devient le nombre hexadécimal 6 après un décalage à droite de 2 bits ?

$6 \gg 2 = 1$

❖ Que devient en hexadécimal, le nombre hexadécimal A7 après un décalage à droite de 2 bits ?

$A7 \gg 2 = 29$

❖ Que devient le nombre hexadécimal 101 après un décalage à droite de 4 bits ?

$101 \gg 4 = 10$

❖ Que devient le nombre hexadécimal 15A après un décalage à droite de 1 bit ?

$15A \gg 1 = AD$

❖ Masques

❖ Comment ne garder que les 4 bits de gauche du nombre AF_{16} ?

$(AF_{16} \& F0_{16}) \gg 4 = A_{16}$

❖ Comment ne garder qu'un bit sur 2 du nombre $A5_{16}$ à partir de la droite, et mettre les autres à 0 ?

$(A5_{16} \& 55_{16}) = 05_{16}$

❖ Comment ne garder que les 4 bits de droite du nombre FA_{16} ?

$$(FA_{16} \& 0F_{16}) = A_{16}$$

- ❖ Comment mettre à 0 les 4 bits de gauche du nombre AF_{16} ?

$$(AF_{16} \& 0F_{16}) = F_{16}$$

- ❖ Comment mettre à 1 les 4 bits de gauche du nombre AF_{16} ?

$$(AF_{16} | F0_{16}) = FF_{16}$$

- ❖ Comment mettre à 1 les 4 bits de gauche avec le nombre AF_{16} et le masque $0F_{16}$?

$$(AF_{16} \& !0F_{16}) = FF_{16}$$

- ❖ Comment mettre à 0 les 4 bits de droite avec le nombre $A5_{16}$ et le masque $0F_{16}$?

$$(A5_{16} \& !0F_{16}) = A0_{16}$$

❖ Programmes

○ Exercice 1

Donnez le résultat du programme suivant :

```
#include<stdio.h>
int main()
{
    int a,b;
    int c = 6, d = 1;
    a = c&d;
    b = c|d;
    printf("a=%d ; b=%d ;", a, b);
    return 0;
}
```

Solution :

& c'est le "ET binaire bit à bit",

| c'est le "OU binaire bit à bit",

Le programme va donc nous faire un ET logique bit à bit entre 6 : 0100 et 1 : 001 :

soit 0 : 000 pour a et

7 : 111 pour b

Donc l'affichage sera : 0 7

○ Exercice 2

Donnez le résultat du programme suivant :

```
#include<stdio.h>
int main()
{
    int a,b;
    a = 0b11001101;
    b = 0b00001111 & a;
    printf("a=%d ; b=%d ;", a, b);
    return 0;
}
```

Solution :

a = 0b11001101;
b = 0b00001111 & a;
ceci donne en résultat 0b00001101
Soit en décimal -51 et 13.

○ **Exercice 3**

Donnez le résultat du programme suivant :

```
#include<stdio.h>
int main()
{
    int a,b;
    int c = 6, d = 1;
    a = c && d;
    b = c || d;
    printf("a=%d ; b=%d ;", a, b);
    return 0;
}
```

Solution :

Concernant les opérateurs && et ||, ils ne peuvent produire que deux résultats différents : 0 (faux) ou 1 (vrai).

L'opérateur && produit 0 si le premier opérande est nul.

L'opérateur && produit 1 si la première et les seconds opérandes sont non nuls.

L'opérateur || produit 0 si la première et les seconds opérandes sont nulles.

L'opérateur || produit 1 si le premier opérande est non nul.

On obtient donc a=1 et b=1 !